REMARKS

Claims 1-6, 8-15, and 17-23 are pending in this application. No matter is added by the amendment submitting the serial numbers of cross-referenced applications.

The Office Action mailed April 8, 2004 objected to the disclosure and rejected claims 1-6, 8-15, and 17-23 as obvious under 35 U.S.C. § 103 based on *O'Donnell et al.* (US 6,480,877). This rejection is respectfully traversed because *O'Donnell et al.* fails to teach or suggest the features of the claims.

For example, all the claims recite "computer-implemented steps" or a "computer-readable medium bearing instructions" that include "generating a layout" (or a "plurality of layouts") "in a high-order language." Although *O'Donnell et al.* shows sample C programming language code, *O'Donnell et al.* fails to describe how that code is generated, much less by "computer-implemented steps" or by execution of instructions on a "computer-readable medium" in the manner claimed.

Indeed, the details in how that code is generated are not the focus of the reference. Rather, *O'Donnell et al.* is directed to a way to identify orphan computer processes. As explained in col. 2:10-30, "[o]rphan computer processes are computer processes that are active or otherwise consume system resources on a computer system but that have no user that owns the processes." Processes can become orphans in various ways, e.g. when a network connection with a user is lost, but some non-user owned processes are really legitimate system, administration, SQL processes that should not be identified as orphans. Accordingly, *O'Donnell* describes the use of an `exclude()` function that "skips the current process if it is in a class or is otherwise identified as a process that is not an orphan" (col. 7:36-38). The implementation for the `exclude()` function is found in the code section of cols. 5-6, in which the `exclude()`

function takes a parameter that is a pointer to a structure describing the status of a particular process (`struct pst_status *proc`). As explained in col. 9:52-63, the declaration of the `pst_status` structure is found in a header file supplied by a third-party vendor.[1] Accordingly, since the declaration of the `pst_status` structure was supplied by a third party, it is not surprising that *O'Donnell et al.* give any details as to how that declaration was generated.

Even if, upon further search or consideration, a reference were to be found showing the generation of a layout in a high-order language by "computer-implemented steps" or by execution of instructions on a "computer-readable medium," *O'Donnell et al.* still does not describe other features of the claims. For instance, independent claim 1 recite:

> generating a layout for the object in a high-order language based on the definition of the object and the size and alignment of the one or more primitive types.

Case law requires claim language referred to by antecedent basis to be read on the same thing, but in the statement of the rejection the Office Action read the "object" limitation on two different and incompatible portions of *O'Donnell et al.* First the Office Action, p. 3, equated the recited object with the `proc` parameter, but then switched to the `exclude()` function (emphasis in **bold** added):

> generating a layout (struct pst_status *proc, line 32 of code table columns 5-6) for the **object (proc, line 21 of code table column 5-6)** in a high-order language (C programming language, lines 9-10 column 6) based on the definition of the **object (int exclude(proc)**, line 31 of code table columns 5-6) and the size and alignment (sizeof(struct pst_dynamic), line 26 of code table columns 7-8) of the one or more primitive types (int exclude(proc), line 31 of code table columns 5-6).

Thus, the statement of the rejection reads "object" in the claim language on both a parameter declaration (`struct pst_status *proc`) and a function declaration (`int exclude(proc)`). However, the parameter `proc` is not same thing as the `exclude`

[1] A sample declaration for the pst_status structure apparent from that vendor is found in Appendix A, cols. 19–22.

function. Moreover, the function declaration `int exclude(proc)` is also not a "definition of the object," since it declares the function `exclude()`—it does not define the parameter `proc`.

Furthermore, the function declaration `int exclude(proc)` also seems to do double-duty in the rejection, also being equated to the recited "one or more primitive types." A specific function (the `exclude()` function) is not a primitive type, however. In addition, the `sizeof(struct pst_dynamic)` does not give the size or alignment of `int exclude(proc)` nor even the `proc` parameter, which is a pointer to a `struct pst_status`, not `struct pst_dynamic` of a different structure. In fact, the C `sizeof` operation does not even yield an alignment.

In regard to independent claims 8 and 21, they were rejected "for the same reasons as claim 1 above" and "for the same reasons as claims 8-9 and 19 above." Although the limitations in independent claims 8 and 21 differ from those of claim 1, they both recite the element "the object" at least twice. Since these claims were rejected "for the same reasons," the rejection of claims 8 and 21 are traversed similarly—because recitations of "the objects" are read on different elements.

The remaining dependent claims and corresponding computer-readable medium claims and are allowable for at least the same reasons as given above and are individually patentable on their own merits.

Therefore, the present application, as amended, overcomes the objections and rejections of record and is in condition for allowance. Favorable consideration is respectfully requested. If any unresolved issues remain, it is respectfully requested that the Examiner telephone the undersigned attorney at 703-425-8516 so that such issues may be resolved as expeditiously as possible.

Respectfully Submitted,

DITTHAVONG & CARLSON, P.C.

[illegible]

Stephen C. Carlson
Attorney/Agent for Applicant(s)
Reg. No. 39929

7/8/2004
Date

10507 Braddock Rd
Suite A
Fairfax, VA 22032
Tel. 703-425-8516
Fax. 703-425-8518